SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Marin Lukas

Zagreb, 2013.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Mladen Crneković, dipl. ing.

Student:

Marin Lukas

Zagreb, 2013.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru te svima koji su mi pomogli tijekom studija.

Marin Lukas

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:

procesno-energetski, konstrukcijski, brodostrojarski i inženjersko modeliranje i računalne simulacije

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: Mat. br.:

Naslov rada na hrvatskom jeziku:

Naslov rada na engleskom jeziku:

Opis zadatka:

Zadatak zadan: Rok predaje rada: Predviđeni datumi obrane:

Zadatak zadao: Predsjednik Povjerenstva:

Prof. dr. sc. Igor Balen

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS PRILOGA

Prilog 1. Programski kod za izvršavanje na PC računalu

Prilog 2. Programski kod za izvršavanje na računalu RaspberryPi

Prilog 3. CD sa radom u digitalnom obliku, videosnimkom rada robota i progarmskim kodovima.

POPIS OZNAKA

Oznaka	Jedinica	Opis
Vt	m/s	Translacijska brzina robota
ω	Rad/s	Kutna brzina robota
Kp	-	Konstanta P regulatora
Xm	Pixeli	Mjerena veličina s kamere
Xv	Pixeli	Vodeća veličina
Φ	Rad	Kut objekta u ondnosu na os robota
d_p	Mm	Širina polovne fidnog polja
S	Mm	Širina vidnog polja na udaljenosti D
D	Mm	Udaljenost z akalibraciju kamere
Kc	-	Kostanta kalibracije kamere
Dm	Mm	Polovina širine vidnog polja na udaljenosti Xm
d	Mm	Izmjerena udaljenost lopte od robota
Xe	Pixeli	Signal pogreške

SAŽETAK

U ovom radu detaljno je opisano rješavanje problema loptanja robota eMIR. Loptanje je raščlanjeno na osnovna gibanja i prema tome je razrađena strategija vođenja robota do lopte. Robot eMIR (educational Mobile Inteligent Robot) je edukacijski mobilni robot razvijen i izrađen na Katedri za strojarsku automatiku Fakulteta strojarstva i brodogradnje u Zagrebu. Za pronalaženje i praćenje lopte korištena je standardna web-kamera, a čitav program izvršava se na uobičajenom prijenosnom računalu (Netbook). Program je napisan korištenjem programskog jezika Python. Komunikacija između robota i računala odvija se putem Bluetooth RS232 modula. U drugom dijelu rada na robota je ugrađeno računalo RaspberryPi na kojem je pokrenuta modificirana verzija razvijenog programa.

Ključne riječi: Mobilni robot, vizijski servo sustav, vizijski sustav, loptanje, IBVS, eMIR Raspberry Pi, Python.

SUMMARY

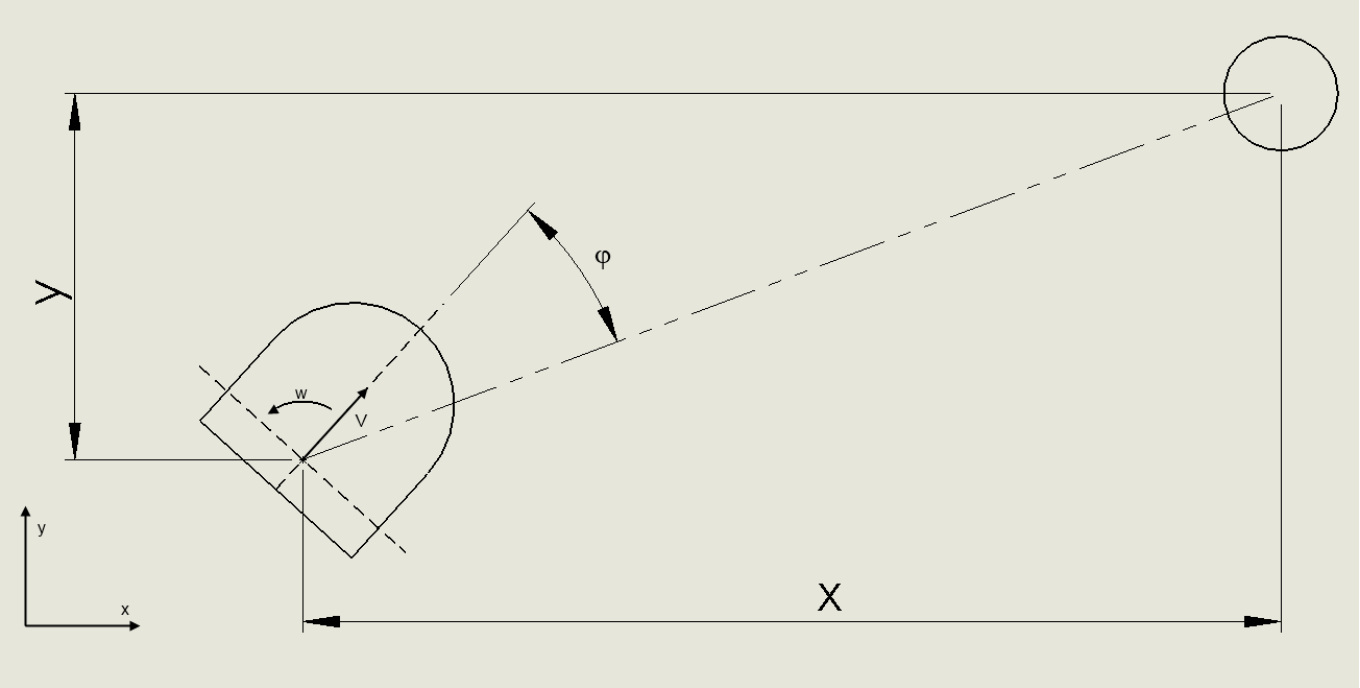
In this paper, a ball playing problem for mobile robot eMIR is solved. Playing with a ball is divided into core motion and therefore strategy of moving with a ball is elaborated. Robot Emir (Educational Mobile Intelligent Robot) is an educational mobile robot developed and manufactured at the Faculty of Mechanical Engineering and Naval Architecture in Zagreb. To detect and track the ball standard webcam is used. The entire program is executed on a typical laptop (netbook). The program is scripted in Python programming language. Communication between robot and computers is done through Bluetooth RS232 module. In the second part of the work on the robot is controlled by onboard computer RaspberryPi that is running a modified version of the developed program.

Keywords: Mobile robot, visual servoing, robot playing with a ball, IBVS, Raspberry Pi, eMIR, Python, Camera tracking

1. UVOD

Iako je loptanje u čovjekovom svijetu jedna od prvih igara koju naučimo i predstavlja vrlo jednostavnu radnju, u svijetu robotike udariti loptu slijediti istu te je ponovno udariti, predstavlja složeniji zadatak. U ovom radu opisano je rješavanje problema loptanja korištenjem mobilnog robota eMIR. Kako gibanje lopte nije unaprijed poznato položaj lopte potrebno je odrediti nekim senzorom te prema njemu voditi robota do lopte. Ovakvim promatranjem loptanje se svodi na probleme presretanja i slijeđenja koji se pojavljuju u mnogim praktičnim primjenana robota kao što su: robotske igre, automatizirani nadzor, automatizacija kućanskih poslova te razne spasilačke, vojne i druge istraživačke primjene. U literaturi je moguće pronaći razne pristupe rješavanju ovih problema, no većina tehnika kao glavni senzor za određivanje položaja objekta koji treba presresti (lopte) koristi kameru. Kamera se može nalaziti na robotu ili nekom fiksnom mjestu u prostoru s kojega je moguće vidjeti robota sa čitavom radnom okolinom. Isto tako neki autori koriste pomičnu, a neki fiksnu kameru. U ovom radu korištena je fiksna kamera smještena na robotu. Računalo i kamera čine vizijski sustav pomoću kojega je moguće uspostaviti vizijsku povratnu vezu [1]. Sustavi sa vizijskom povratnom vezom nazivaju se vizijski servo sustavi [1] i pojavljuju se početkom 1980-tih. kod realizacije vizijskih servosustava postoje dva uobičajena osnovna pristupa [2] IBVS (*Image Based Visual Servoing*) ili Vizijski sustavi temeljeni na obradi slike i PBVS (*Position Based Visual Servoing*). Pristup IBVS direktno koristi veličine sa slike u regulacijskoj petlji za regulaciju stupnjeva slobode gibanja robota, dok se kod pristupa PBVS prvo sa slike određuje položaj objekta u koordinatnom sustavu robota pa se te veličine koriste u regulacijskoj petlji. U ovom radu korišten je IBVS pristup. Kao platforma za ispitivanje računalnog programa i strategije gibanja odabran je robot eMIR koji je konstruiran i izrađen na Fakultetu strojarstva i brodogradnje u Zagrebu. Za upravljanje robotom odabrano je uobičajeno osobno prijenosno računalo (Netbook) s ugrađenom kamerom te operativnim sustavom Windows. U drugom dijelu razvijeni upravljački program implementiran je na računalo Raspberry Pi kako bi se pokazala jednostavna implementacija postojećih rješenja pisanih u programskom jeziku Python na mikrokontrolere iz porodice ARM s operativnim sustavom Linux. Ovakva konfiguracija računala nije namjenjena industrijskoj ili bilo kakvoj drugoj primjeni u robotici, no odabrana je kako bi se pokazalo što je moguće napraviti s uređajima koje većina nas posjeduje i ne zahtjevaju nikakve posebne investicije.

2. STRATEGIJA VOĐENJA ROBOTA DO LOPTE



Slika 1. Prikaz robota i lopte u ravnini

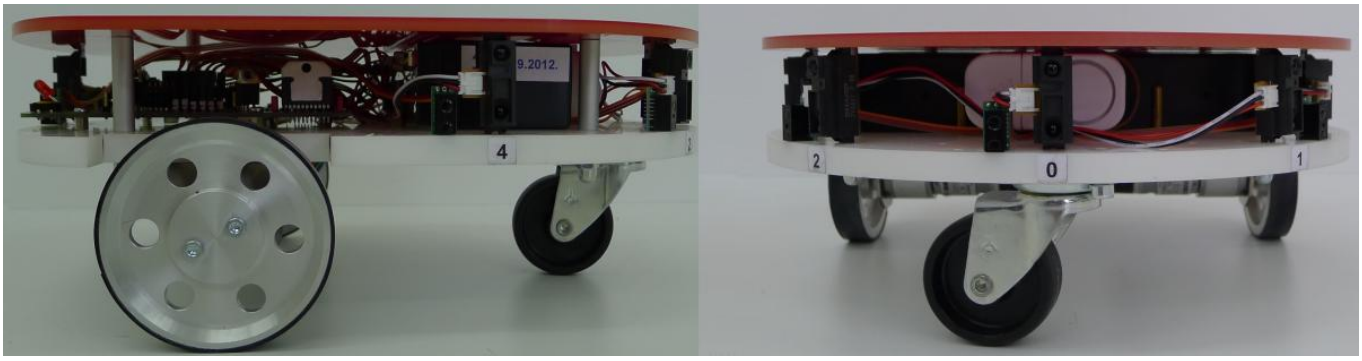
Kako bismo doveli robota u kontakt s loptom koja se nalazi na nekoj udaljenosti u horizontalnoj ravnini potrebno je razviti strategiju vođenja. S obzirom da između robota i lopte ne postoje nikakve prepreke dovoljno je osigurati gibanje robota po pravcu prema lopti. Strategiju vođenja robota do lopte možemo podijeliti u 4 faze :

1. **Pronalazak i identifikacija lopte**
2. **Okretanje prema lopti**
3. **Gibanje prema lopti**
4. **Udaranje (sudar) lopte**

Prva faza je pronalazak i identifikacija lopte određene boje u vidnom polju kamere. Robot u ovoj fazi rotira konstantnom kutnom brzinom $\boldsymbol{\omega}$ i faza traje sve dok se u vidnom polju kamere ne pojavi objekt zadane boje. Druga faza je regulacija kutne brzine u svrhu minimizacije kuta objekta u odnosu na uzdužnu os robota $\boldsymbol{\varphi}$ do zadane granice. Tijekom druge faze kutna brzina robota postupno se smanjuje dok je translacijska brzina jednaka nuli. U trećoj fazi koja počinje kada je kut objekta u odnosu na uzdužnu os robota unutar zadanih granica, započinje translatorno gibanje robota konstantnom brzinom $\boldsymbol{v}$. Kutna brzina se tijekom treće faze regulira pomoću povratne veze sa kamere kako bi se kut $\boldsymbol{\varphi}$ držao što je moguće bliže iznosu 0°, a time i robot gibao prema lopti. Kada robot priđe lopti na udaljenost manju od 10cm započinje četvrta faza u kojoj se translacijska brzina v poveća na maksimalnu vrijednost u vremenskom intervalu od 0.5s kako bi robot udario loptu. Nakon toga zaustave sva gibanja robota u intervalu od 1s. Poslije sudara s loptom sekvenca gibanja se ponavlja od prve do četvrte faze te se tako dobija dojam da se robot lopta. U daljnjem tekstu biti će detaljno opisan način ostvarivanja gibanja u svim fazama. [Slika 1] prikazuje položaj lopte i robota u horizontalnoj ravnini.

3. OPIS ROBOTA educational Mobile Inteligent Researcher

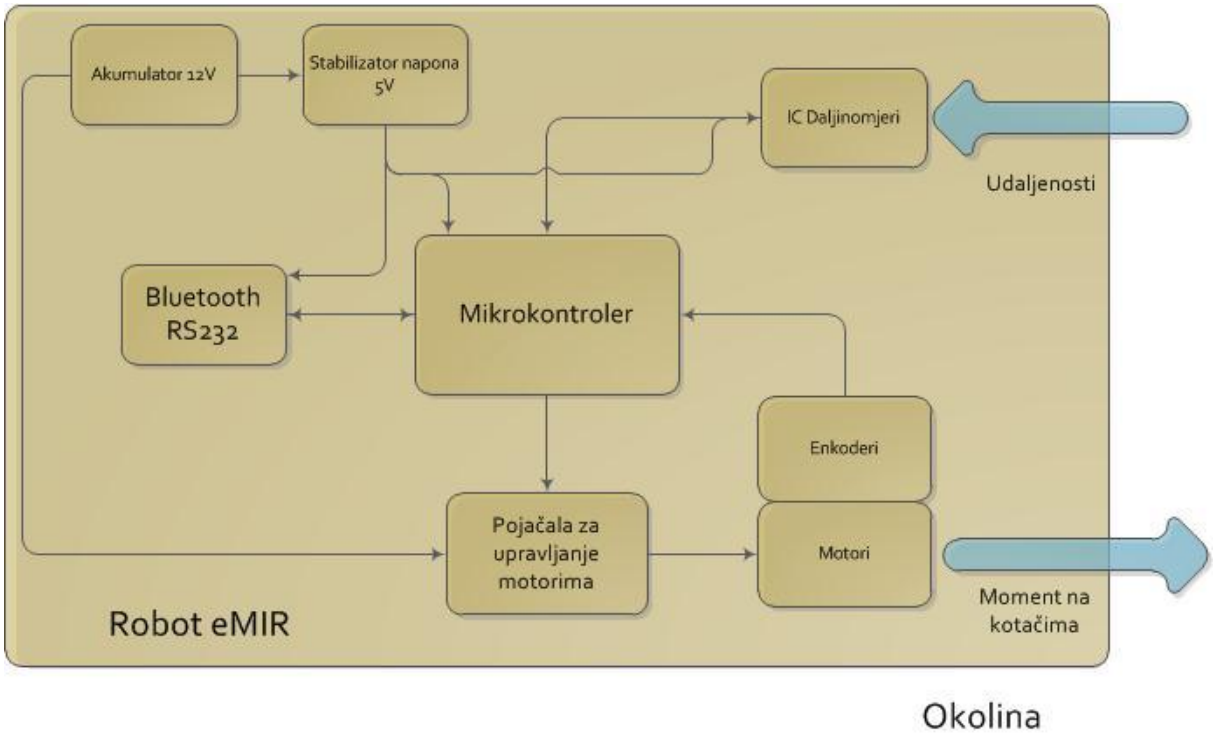
3.1. MEHANIČKA KONSTRUKCIJA ROBOTA



Slika 2. Mehanička konstrukcija robota

Robot eMIR konstruiran je i izrađen na Katedri za strojarsku automatiku Fakulteta strojarstva i brodogradnje u Zagrebu. Robot je diferencijalne strukture, što omogućava okretanje oko vertikalne osi u mjestu te pojednostavnjuje konstrukciju. Dva stražnja kotača pogone se neovisnim istosmjernim motorima s planetarnim reduktorima. Enkoderi su inkrementalnog tipa i smješteni su na vratilo motora. Motori i kotači postavljeni su na ploču od PMMA (poli metil metakrilat) te pričvršćeni vijcima. Kotači robota izrađeni su od aluminijske legure te obloženi elastomernom oblogom kako bi se povećalo trenje između kotača i podloge te time smanjilo proklizavanje. Iznad temeljne ploče nalazi se pokrovna ploča koja pridržava senzore za mjerenje udaljenosti raspoređene na vanjskoj strani robota i štiti upravljačku elektroniku od vanjskih utjecaja. Na pokrovnu ploču moguće je smjestiti dodatnu opremu. U ovom slučaju na pokrovnoj ploči nalazi se računalo (Netbook) koje upravlja robotom. [Slika 2] prikazuje robota iz dvije projekcije.

3.2. UPRAVLJAČKA JEDINICA

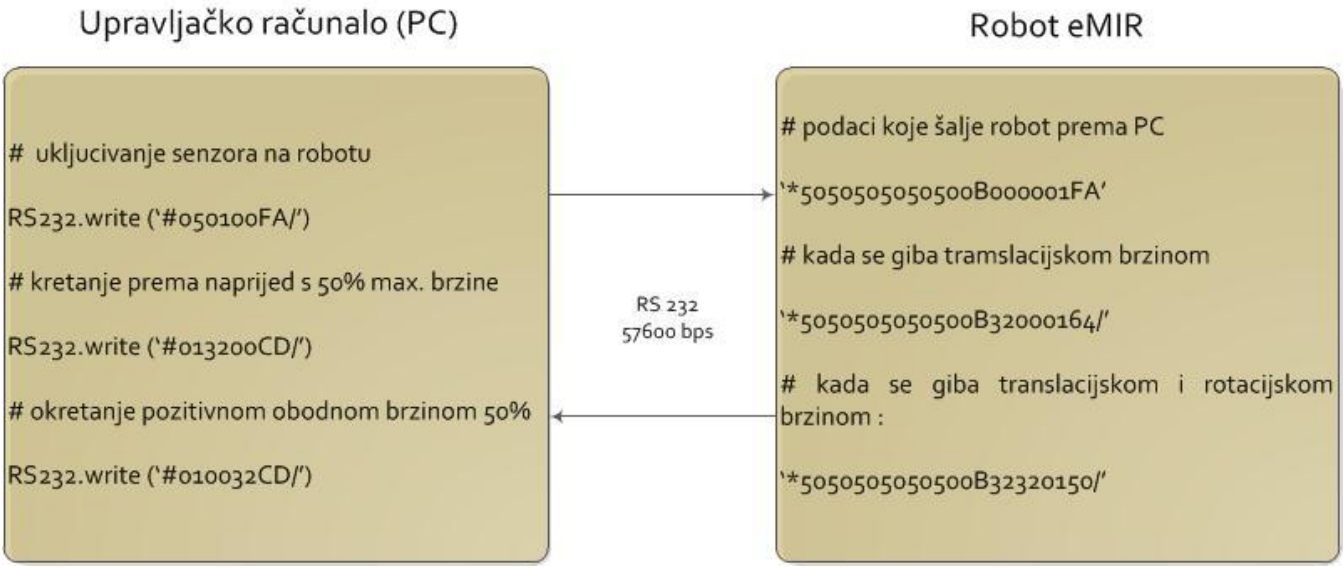


Slika 3. Blok dijagram upravljačke jedinice

Upravljačka jedinica robota eMIR također je konstruirana i izrađena na katedri za strojarsku automatiku fakulteta strojarstva i brodogoradnje u Zagrebu. Glavni dio upravljačje jedinice je mikrokontroler iz porodice 8051[3] u kojem se nalazi program za regulaciju brzina translacije i rotacije. Motorima se upravlja pomoću širinsko impulsne modulacije (PWM), a snaga se motorima dovodi preko integriranih krugova sa ugrađenim H-mostovima. Enkoderi motora spojeni su direktno na mikrokontroler. Komunikacija između glavnog računala i upravljačke jedinice odvija se putem serijske veze RS232. Kako bi se izbjeglo spajanje kablova na upravljačku jedinicu komunikacija se uspostavlja putem Bluetooth modula instaliranog na pločicu. Za mjerenje udaljenosti između robota i ostalih objekata u okolini koriste se analogni infracrveni daljinomjeri Sharp. Analogni signal pretvara se u digitalni korištenjem analogno digitalnog pretvornika te se nakon toga u digitalnom obliku proslijeđuje na mikrokontroler.

Ovakvim sustavom za detekciju prepreka moguće je uspješno mjeriti udaljenosti prepreka od robota u rasponu 0-80cm. Čitav sustav napaja se iz olovnog akumulatora napona 12V i kapaciteta 2Ah što robotu u prosjeku daje autonomiju od tri sata. Osim pomoću Bluetootha serijsku vezu moguće je uspostaviti putem kablova, a čitav program u upravljačkoj jedinici je moguće mijenjati prema potrebi korisnika. Informacijski i energetski dio nalaze se na istoj tiskanoj pločici što nije u skladu sa industrijskom praksom, ali u ovom slučaju ne stvara probleme zbog relativno male snage motora. [Slika 3] prikazuje blok shemu upravljačke jedinice robota.

3.3. OPIS KOMUNIKACIJSKOG PROTOKOLA



Slika 4. Primjer komunikacijske sekvence

S obzirom da se unutar upravljačke jedinice nalazi PI regulator kutne i translacijske brzine korisniku izvana nije potreban pristup širinsko impulsnoj modulaciji i signalima sa enkodera, već se robotu putem serijske veze šalju reference kutne brzine **ω** i translacijske brzine **vt**. Komunikacija se odvija brzinom od 57600BPS što je dovoljno s obzirom da je interval prijema podataka 0.1s. Svaka naredba poslana robotu započinje znakom „#“, a završava znakom „/“. Između navedenih znakova nalaze se informacije o modu rada, translacijskoj i

kutnoj brzini te kontrolna vrijednost (Checksum). Sve brojčane vrijednosti zadaju se u heksadekatskom sustavu. Osim kontrole gibanja robotu je moguće slati naredbe za uključivanje senzora. Kada robot primi naredbu za uključivanje senzora, upravljačka jedinica računalu šalje paket podataka u kojem se nalaze vrijednosti daljinomjera, trenutna kutna i translacijska brzina te napon baterije. Sve brojčane vrijednosti također su poslane u heksadekatskom brojevnom sustavu. Naredbe te pripadajuće radnje prikazane su u [Tablici 1.] preuzetoj iz [3] Primjer komunikacijske sekvence prikazuje [Slika 4].

Tabela 1. Naredbe za upravljanje robotom

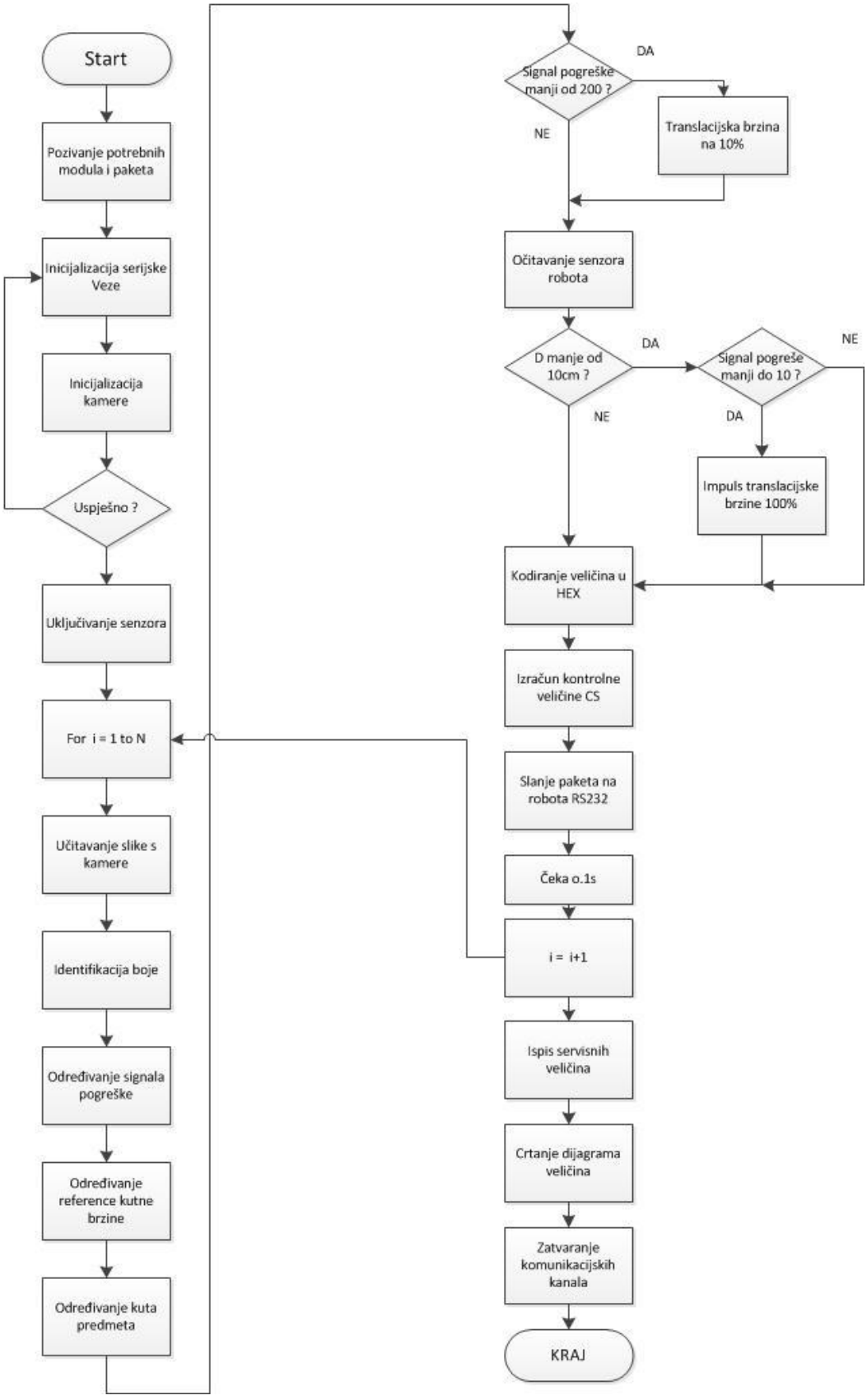
NN	Command	Description
0	# 00 00 00 00 /	Stop the robot and exit from a current task.
1	# 01 vv rr CS /	Move the robot with translation speed *vv* and rotation speed *rr* (–100 to +100%).
17	# 11 PP 00 CS /	Define returned information package PP (0, 1 or 2).
19	# 13 tt 00 CS /	Turn on internal horn for time tt tenths of seconds.
255	# FF 00 00 01 /	Turn off the robot.

4. RAČUNALNI PROGRAM ZA RJEŠAVANJE PROBLEMA LOPTANJA

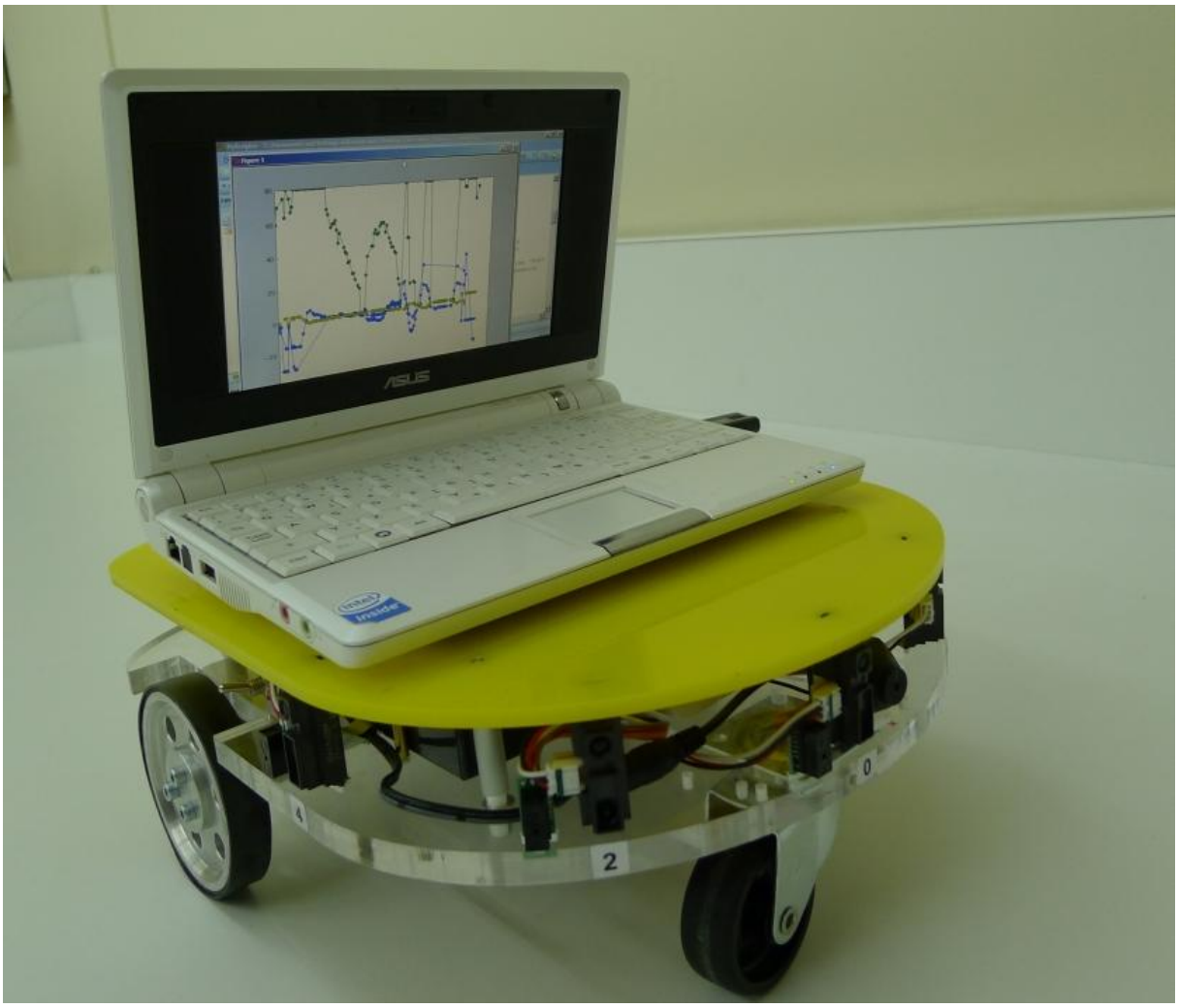
Opisana strategija gibanja na robotu se izvršava putem programskog koda u računalu. Program je napisan korištenjem programskog jezika Python 2.7 i razvojnog okruženja Py scripter. Navedeni programski jezik odabran je zbog svoje objektne strukture koja omogućava jednostavan pristup kameri i podržava velik broj vanjskih modula i paketa obradu slike, serijsku komunikaciju i ostatale radnje potrebne za vođenje robota. Osim toga programiranje u jeziku Python sastavni je dio nastavnog programa na smjeru Mehatronika i robotika Fakulteta strojarstva i brodogradnje što će olakšati budući rad na ovoj problematici. Kao i strategija gibanja program je također podjeljen u nekoliko osnovnih dijelova :

1. **Pozivanje potrebnih modula**
2. **Inicijalizacija kamere i serjske veze**
3. **Programska petlja za izvršavanje zadatka**
4. **Ispis servisnih podataka**
5. **Zatvaranje komunikacijskih kanala prema robotu i kameri.**

Za prikupljanje podataka sa kamere i analizu slike korišten je modul PYGAME koji u sebi sadrži neke dijelove poznate programske biblioteke za obradu slike OPEN CV. Pygame je paket za razvoj računalnih igara, a u ovom slučaju se pokazao dobar zbog mogućnosti jednostavnog pristupa kameri i detekcije boje te je jednostavniji za korištenje od moćnijeg OPENCV-a. Ispis servisnih podataka uključuje grafički prikaz regulacijskih veličina te zapisivanje poslanih i primljenih naredbi u datoteku. Servisni podaci korišteni su za eliminaciju pogrešaka u programu (debugging) i podešavanje parametara regulatora. Na kraju programa zatvaraju se svi komunikacijski kanali prema kameri, datotekama i serijskoj vezi kako bi drugi programi mogli koristiti navedene kanale nakon prestanka rada s robotom. [Slika 5] prikazuje dijagram toka programa, [Slika 6] prikazuje smještaj računala na robotu s označenim sastavnim djelovima. U daljnjem tekstu biti će detaljno prikazana struktura programa unutar programske petlje za izvršavanje zadatka i način podešavanja regulatora.



Slika 5. Dijagram toka programa za rješavanje problema loptanja



Slika 6. Prikaz smještaja prijenosnog računala na robot eMIR

4.1. Programska petlja za izvršavanje zadatka

Unutar petlje za izvršavanje zadatka neprestano se ponavljaju radnje potrebne za identifikaciju lopte i vođenje robota. Kao što se može vidjeti na dijagramu toka [Slika 5] radi se o konačnoj petlji koja se izvršava zadani broj puta. S obzirom da se unutar petlje nalazi naredba za čekanje iznosa 0.1s zadavanjem broja koraka petlje određujemo vrijeme trajanja

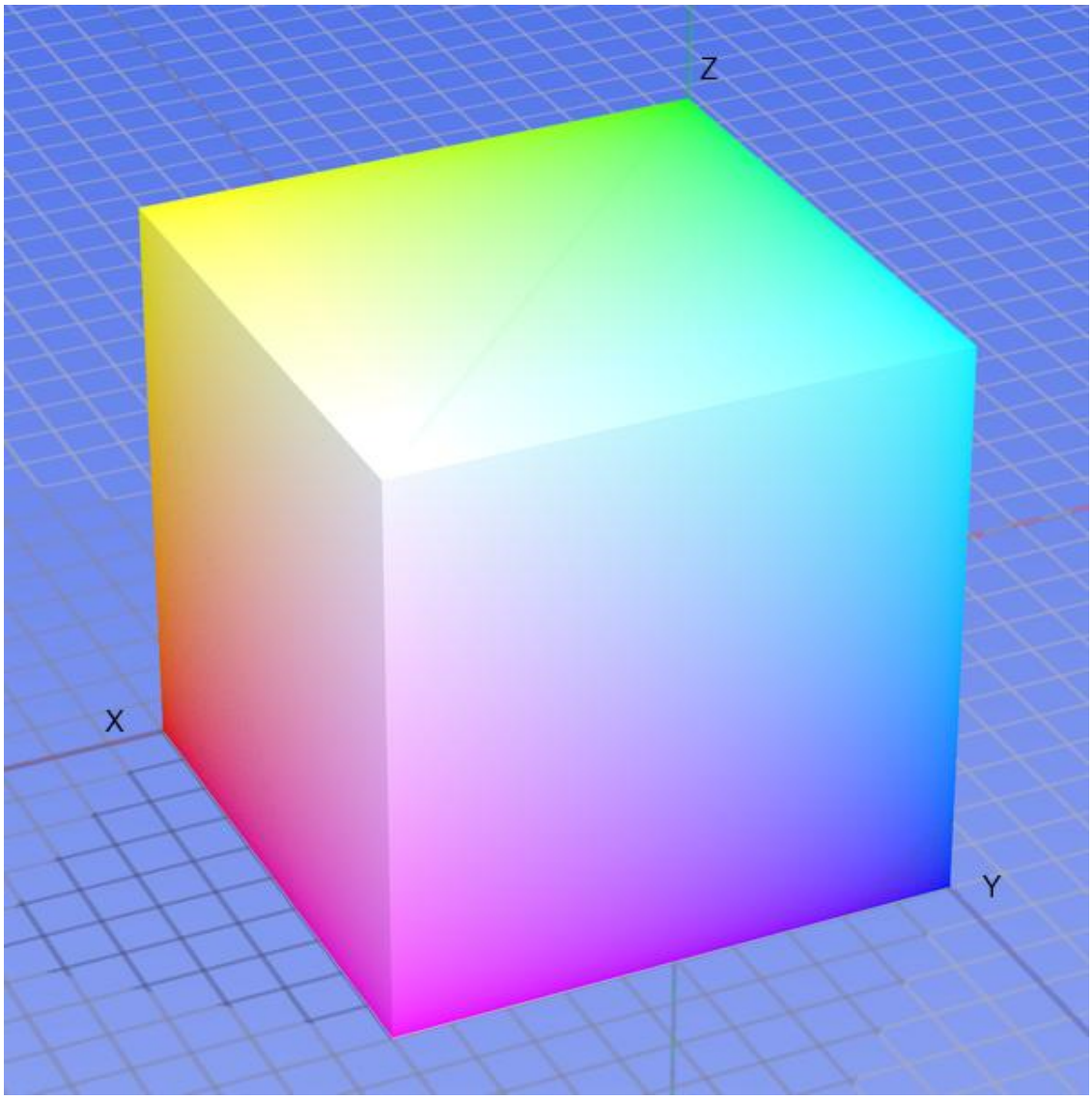
programa tj. vrijeme u kojem će robot izvršavati zadatak udaranja lopte. Radnje unutar petlje možemo podijeliti u šest cjelina:

1. **Korištenje kamere kao senzor položaja lopte**
2. **Regulacija kutne i translacijske brzine**
3. **Očitavanje senzora robota**
4. **Određivanje kuta lopte u odnosu na os robota**
5. **Generiranje udarca**
6. **Slanje podataka prema robotu**

4.2. Kamera kao senzor položaja lopte

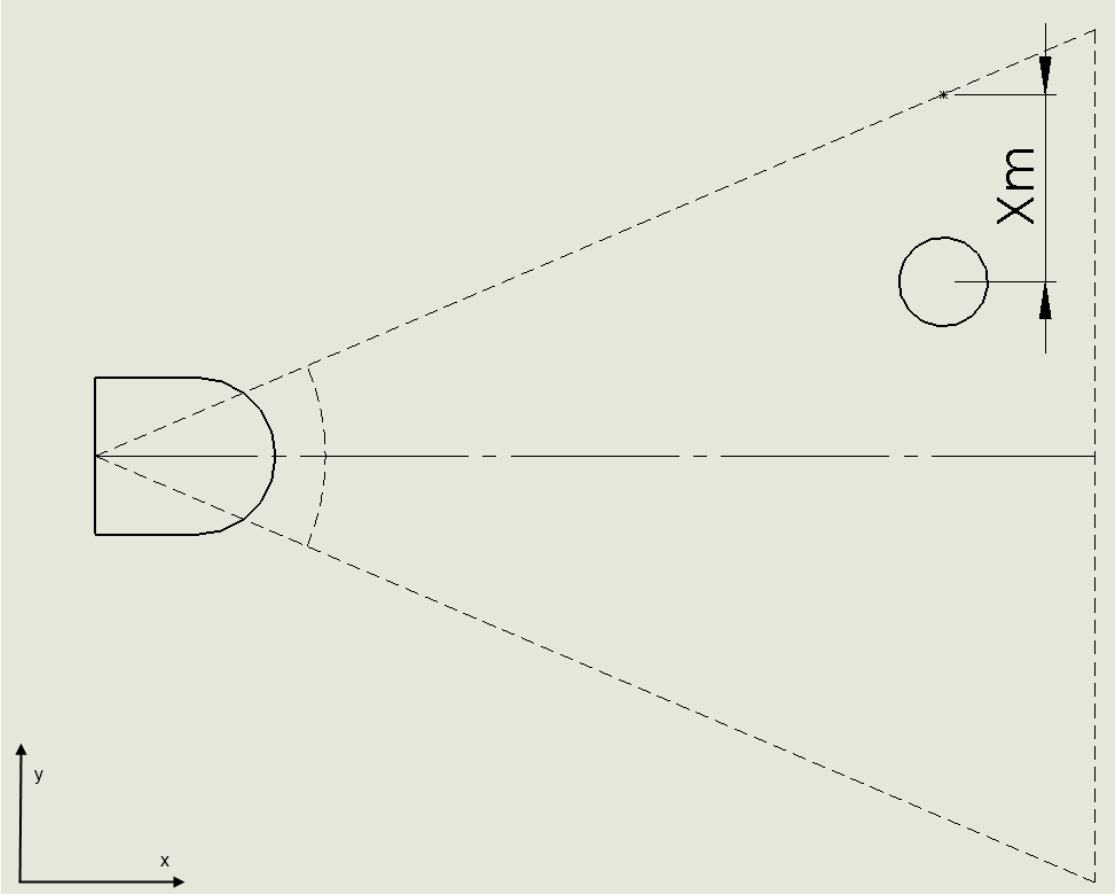
Svaki vizijski servo sustav kao senzor položaja tj. mjerni član u regulacijskom sustavu koristi kameru. U ovom slučaju korištena je web kamera ugrađena na zaslon PC računala (Netbook) Kamera snima rasterske slike u boji frekvencijom 25 fps. Digitalna rasterska slika je matrica dimenzija ***m x n*** gdje *m* i *n* određuju broj piksela u X i Y smjeru tj. veličinu slike. Svaki član matrice je mali cijeli broj koji sadrži informacije o boji navedenog piksela. Boju piksela moguće je prikazati raznim metodama od kojih su najpoznatije HSV (Hue Saturation Value) zapis i RGB (Red Green Blue). Prepoznavanje boja putem HSV zapisa je otpornije na vanjske promjene osvjetljenja i u praksi se češće primjenjuje, no za uvjete ovog rada, na poligonu za mobilne robote Fakulteta strojarstva i brodogradnje dovoljno se dobro pokazalo identificiranje boja pomoću zapisa RGB. Kod RGB zapisa boja se prikazuje pomoću tri cijela broja čija se vrijednost mijenja od 0 do 255. Pa tako kombiniranjem vrijednosti opisujemo sve boje iz

vidljivog spektra. [Slika 7] prikazuje geometrijsku interptretaciju RGB zapisa. Koordinata X pridružena je crvenoj, Y plavoj i Z zelenoj boji. Sa slike možemo vidjeti koje boje odgovaraju kojim RGB vrijednostima. Primjećuje se da se bijela boja prikazuje maksimalnim vrijednostima svih koordinata što odgovara fizikalnom Young-Helmzoltz modelu prema kojem bijelo svijetlo opisujemo kao superpoziciju crvenog, zelenog i plavog. Takva struktura zapisa čovjeku omogućuje lakše snalaženje u prostoru boja od zapisa prema HSV modelu.



Slika 7. Prostorna interpretacija RGB modela

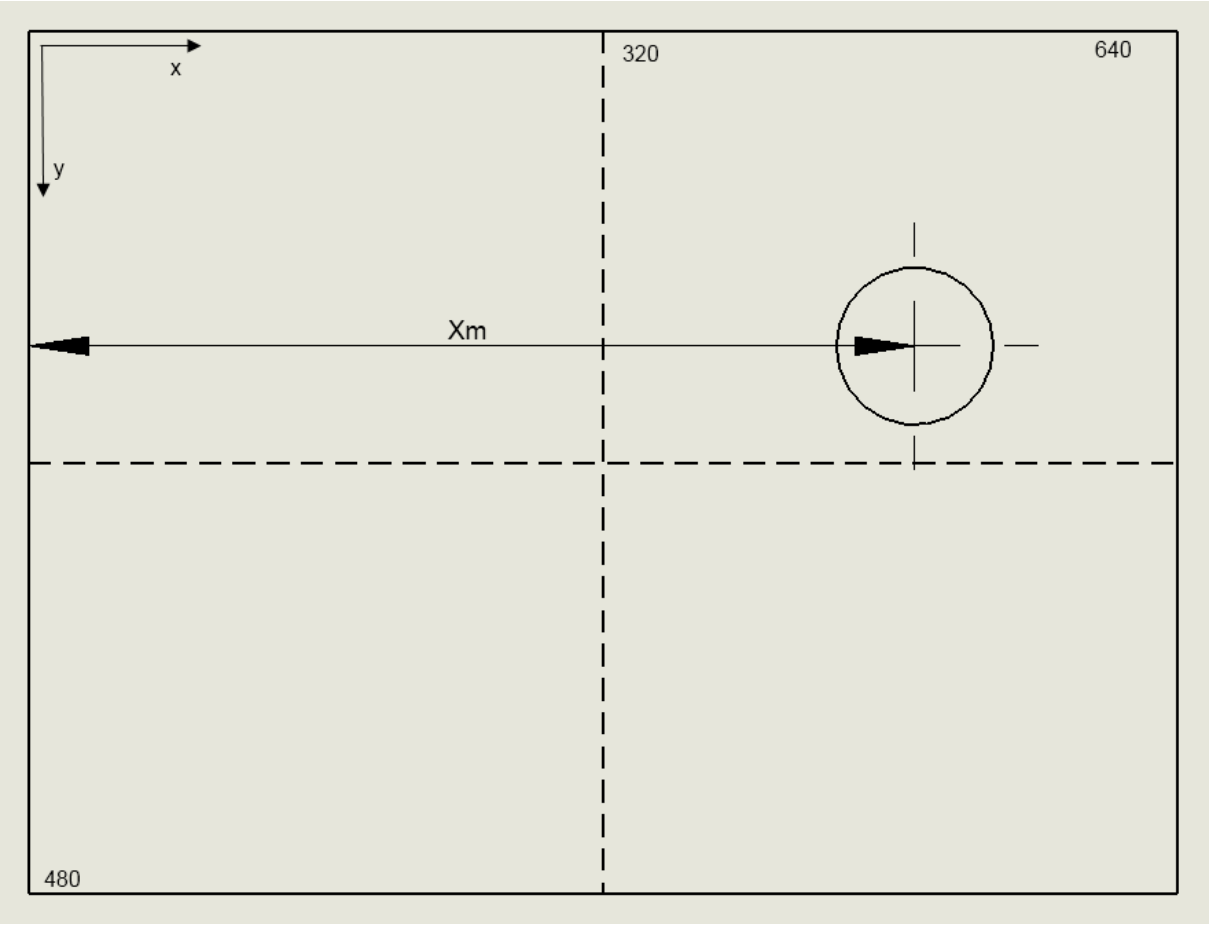
Detektiranje boje rješeno je primjenom RGB modela boje i paketa Pygame. Kao boja koja se prati odabrana je zelena boja RGB vrijednosti R=20 G = 120 B = 20. Nakon učitavanja slike sa kamere na slici se traži skup piksela zadane boje čija se X koordinata težišta uzima za mjerenu veličinu u regulacijskom krugu Xm. RGB vrijednost pixela osim o boji objekta ovisi o intenzitetu i spektru okolišnog osvjetljenja. Da bi bilo moguće pratiti loptu odgovarajuće boje potrebno je zadati raspon vrijednosti R, G i B u kojem se boja još uvijek smatra zadanom (Treshold). Ako je raspon premali, sustav često izgubi loptu, a ako je raspon prevelik, sustav zamjenjuje loptu s drugim predmetima slične boje. Eksperimentalno je određeno da se na zadanom poligonu praćenje najbolje odvija za raspone $\Delta R = 20$, $\Delta G = 50$,



Slika 8. Vidno polje kamere u ravnini

$\Delta B = 20$. Osim detekcije boje za uspješno korištenje kamere kao senzora položaja lopte potrebno je proučiti vezu između položaja piksela u matrici ***m x n*** i stvarnog položaja

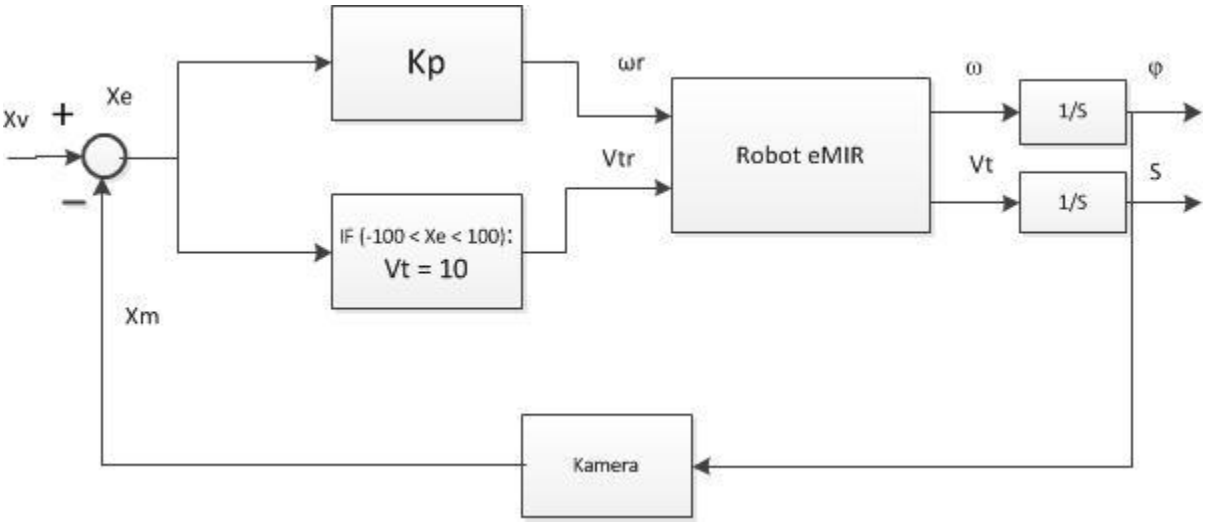
opaženog predmeta u prostoru. Svaka kamera ima vidno polje. Vidno polje kamere je prostor u okolini koji je moguće vidjeti na zaslonu bez pomicanja kamere i u horizontalnoj ravnini je omeđeno jednakokračnim trokutom u čijem se vrhu nalazi kamera. Kut između stranica određen je žarišnom duljinom objektiva kamere i veličinom senzora. U našem slučaju korištena je kamera sa fiksnom žarišnom duljinom. Na [Slika 8] je prikazano vidno polje kamere montirane na robota u horizontalnoj ravnini i objekt u njemu. Promatrani objekt na zaslonu se pojavljuje sa točno određenim koordinatama u pikselima što je prikazano na [Slika 9].



Slika 9. Položaj objekta na zaslonu

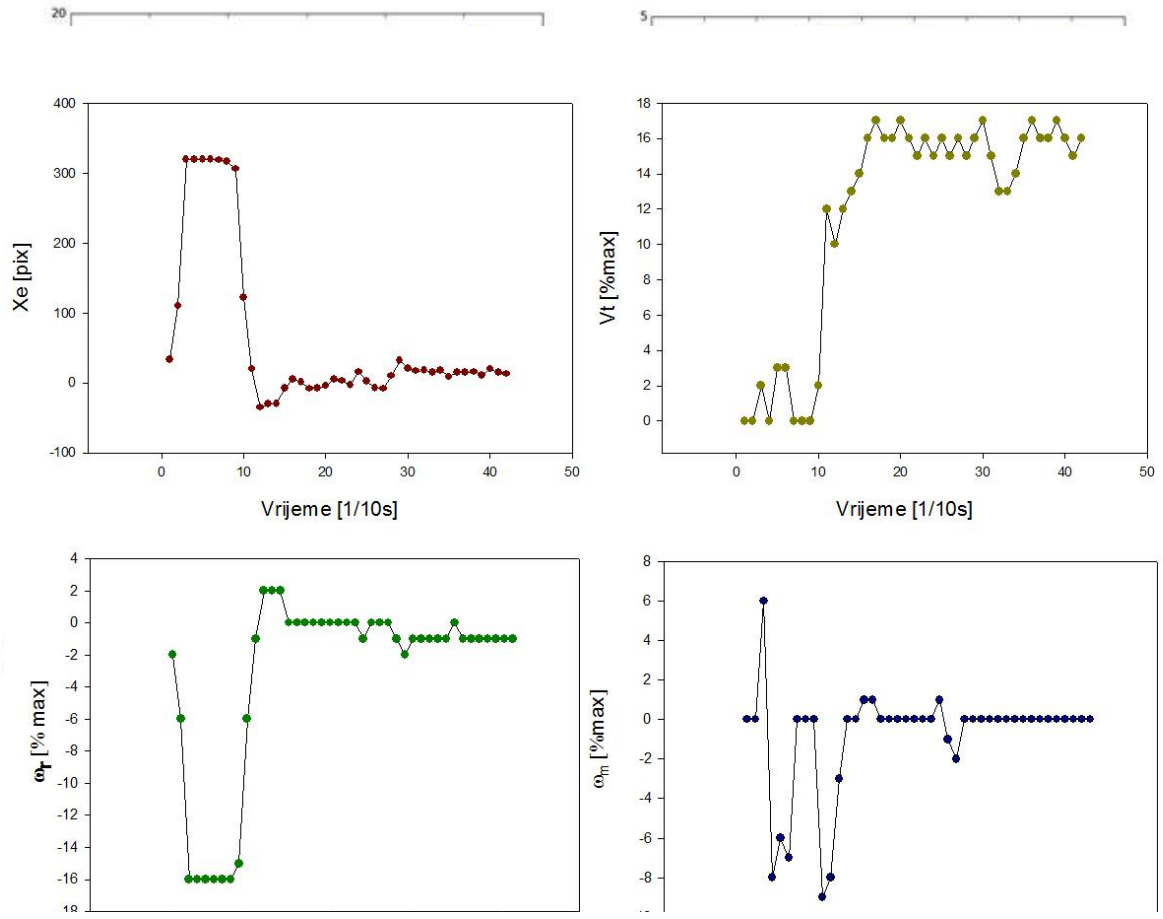
Za korištenu kameru uobičajena veličina zaslona je 640x480 piksela. Kordinate nekog piksela na slici mogu varirati od 0 do 640 u X i od 0 do 480 u Y smjeru. S obzirom da je simetrala vidnog polja postavljena točno na osi robota, ako bismo željeli robota dovesti do željenog predmeta potrebno je robota okretati tako da se centar (težište površine) predmeta nalazi na sredini zaslona, tj. tako da X koordinata odgovara 320 pixela. Iz Y koordinate predmeta na slici mogla bi se odrediti udaljenost do predmeta no u ovom radu to nije korišteno. Za IBVS [2] sustave karakteristično je da se u regulacijsku petlju ulazi direktno sa koordinatama objekata u pikselima, ako bismo željeli koristii PBVS [2] pristup podatke je potrebno preračunavati u stvarne koordinate pomoću kalibracijske funkcije za kameru. Kalibracijska funkcija radi se na temelju mjerenja širine vidnog polja u ovisnosti o udaljenosti od kamere. U ovom radu za praćenje lopte ograničili smo se na IBVS pristup dok je za eksperimentalno određivanje konstante regulacije korištena neka vrsta PBVS pristupa što je opisano u odjeljku Određivanje kuta lopte u odnosu na os robota.

4.3. Regulacija kutne i translacijske brzine



Slika 10. Dinamički model sustava

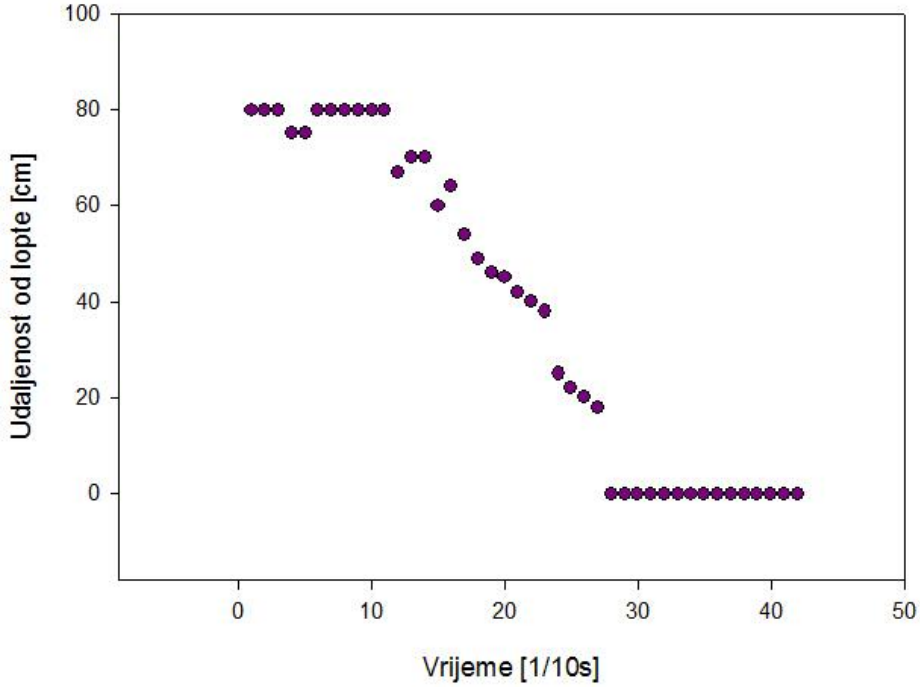
Kako bismo ostvarili tražena gibanja robota potrebno je pomoću opisane kamere i obrade slike napraviti regulaciju kutne i translacijske brzine robota u ovisnosti o položaju lopte na zaslonu kamere. Kao mjerena veličina **Xm** uzima se X koordinate težišta površine zadane boje u pikselima. Kao vodeća veličina postavlja se **Xv** = 320 što odgovara položaju lopte na sredini zaslona. Oduzimanjem mjerenog signala **Xm** od vodeće veličine **Xv** dobivamo signal pogreške **Xe**. Signal pogreške dalje se vodi u P regulator kutne brzine gdje se množi konstantom regulatora **Kp** i nakon toga kao referenca kutne brzine ***ωr*** šalje na robota. Translacijska brzina se ne regulira kontinuirano već se određene vrijednosti signala pogreške **Xe** dobiva vrijednost 10% maksimalne dok je za ostale vrijednosti **Xe** jednaka nuli. Ovime se postiže brže približavanje lopti. Ulaskom lopte u vidno polje signal **Xe** pada te kada padne ispod 100, robot se počne gibati translatorno. U daljnjem gibanju regulator mijenja referentnu



Slika 11. Oscilacije kuta zakreta u ovisnosti o Kp

Slika 12. Grafički prikaz servisnih podataka

kutnu brzinu koja se šalje na robota tako da **Xe** bude čim bliže nuli, a robot se giba translatorno prema lopti. Zbog načina uspostavljanja povratne veze (pomoću slike), **Xe** osim o kutu zakreta robota ovisi i o udaljenosti robota od lopte, što čitav problem čini nelinearnim. Ovu nelinearnost možemo zanemariti jer su oscilacije kuta tijekom gibanja robota prema lopti relativno male. Isto tako mjereni signal se ne može direktno povezati s kutom zakreta pa konstantu **Kp** nije moguće odrediti sintezom regulatora koja je poznata u teoriji automatske regulacije za **P** regulator. Konstanta **Kp** određena je eksperimentalno u nekoliko iteracija. Robot se postavi na udaljenost od 0.8m od lopte okrenut za 180° od smjera prema lopti te se pokrene program s vrlo malom konstantom **Kp**. Za vrlo male konstante **Kp**, dolazak do lopte nije bio moguć zbog prevelikog trenja u reduktorima i njegovog utjecaja na PI regulator kutne brznine robota ugrađen u upravljačku jedinicu robota. Konstanta je postupno povećavana dok sustav nije postao oscilatoran, a rotacija robota prebrza za detekciju boje pomoću kamere. [Slika 11] prikazuje ponašanje kuta zakreta robota za vrijeme gibanja u ovisnosti o konstanti **Kp**. Na kraju je odabrana konstanta **Kp = 0.05** koja se u eksperimentima pokazala optimalnom. Iz servisnih podataka pohranjenih u log-datotekama moguće je nacrtati dijagrame ponašanja regulacijskih veličina **Xv**, **Xe**, **Xm** i **Vt** za opisani eksperiment. [Slika 12] prikazuje dijagrame za **Kp = 0.05** na dijagramu se jasno vidi smanjivanje signala pogreške **Xe**, oscilacije reference kutne brzine **ωr** poslane robotu, nagli skok translacijske brzine **Vt** i odzivi robota snimljeni pomoću informacija o trenutnoj kutnoj i translacijskoj brzini primljenih od robota. [Slika 13] prikazuje dijagram ponašanja udaljenosti do lopte koju mjeri prednji IC senzor. Iz toga možemo vidjeti kako se robot približava lopti. Male oscilacije kod mjerenja udaljenosti koje se primjećuju na dijagramu, uzrokovane su netočnostima u mjerenju IC senzorom, no za ovu primjenu mjerenje je dovoljno pouzdano.



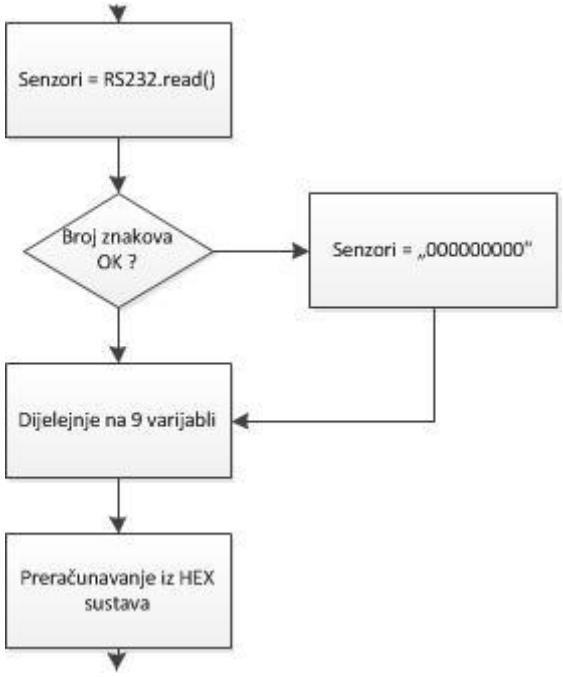
Slika 13. Grafički prikaz udaljenosti od lopte

4.4. Očitavanje senzora robota

Senzori robota očitavaju se putem serijske veze na način koji je opisan u odjeljku o komunikaciji sa robotom. S obzirom da je serijska komunikacija putem bluetootha relativno nepouzdana, potrebno je svaki paket informacija provjeravati. Iako robot u paketu informacija šalje kontrolnu vrijednost CS (Check Sum), u ovomu radu ona nije korištena. Dovljno se pokazalo samo provjeriti broj znakova u paketu te pakete s premalim ili prevelikim brojem znakova odbaciti. U slučaju prijema neispravnog paketa program u servisnu datoteku, na mjesto primljenih podataka ispisuje „000000“ tako da je lako uočiti pogreške u komunikaciji.

Nakon prijema ispravan paket dijeli se na devet veličina od čega prvih šest predstavljaju mjerene vrijednosti IC daljnomjera, sedma vrijednost predstavlja napon baterije, a osma i

deveta kutnu i translacijsku brzinu robota. Dijagram toka dijela programa za očitavanje senzora prikazuje [Slika 14].



Slika 14. Dijagram toka za očitavanje senzora

4.5. Određivanje kuta lopte u odnosu na os robota

Za podešavanje konstante regulatora Kp osim kvalitativnog motrenja oscilacija robota bilo je potrebno mjeriti oscilacije kuta lopte u odnosu na os robota. Odabran je način mjerenja kuta pomoću slike s kamere. Za tu svrhu kamera je kalibrirana. Kalibracija je izvršena mjerenjem širine vidnog polja u ovisnosti o udaljenosti od kamere. Polovina širine vidnog polja u ovisnosti o udaljenosti od kamere zadana je jednadžbom :

$$d_p = \frac{S}{D}\, d \quad [mm]$$

gdje je $\mathbf{d_p}$ širina vidnog polja, **S** mjerena širina vidnog polja na udaljenosti **D**, a **d** udaljenost ravnine u kojoj se nalazi predmet od kamere. Mjerenjem veličina **S** i **D** prema [Slika 15] dobiva se kalibracijska konstanta:

$$Kc = \frac{S}{D} = 0.5917$$

Vrijednosti mjerene veličine na slici Xm uz poznatu ovisnost širine vidnog polja o udaljenosti od kamere, povezujemo sa stvarnom udaljenosti objekta od osi robota relacijom :

$$Dm = \frac{d_p}{320}(Xm - 320) \quad [mm]$$

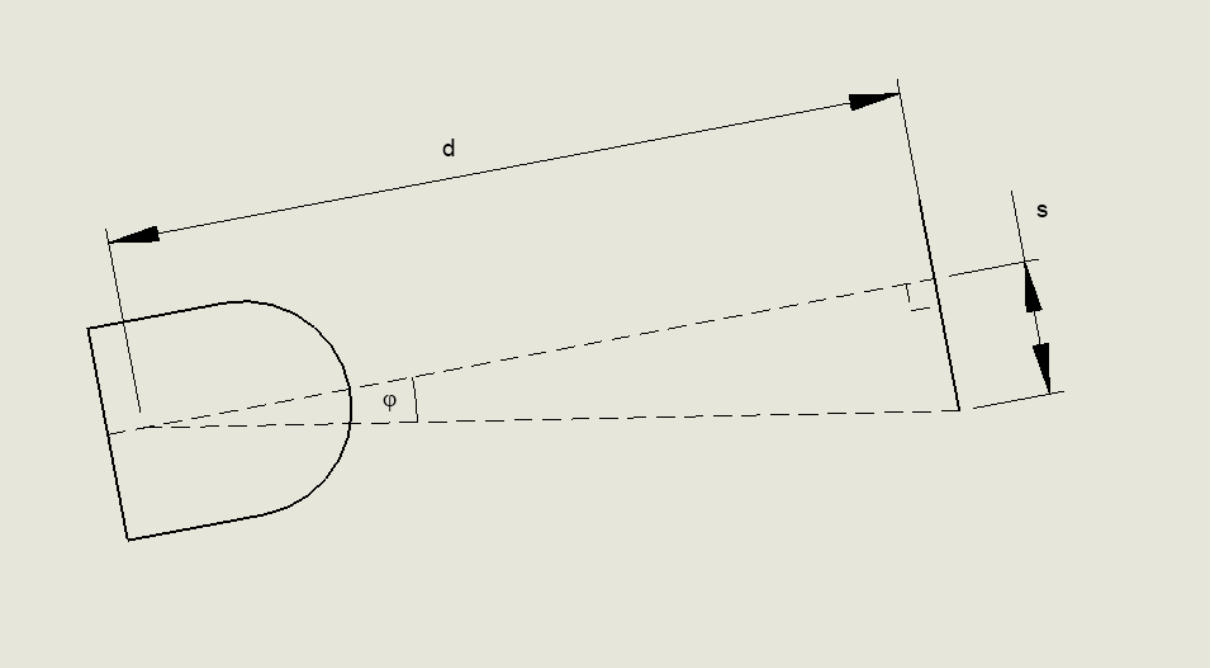
Prema [Slici 15] vidimo da je kut φ uz poznate Dm i d moguće izračunati korištenjem jednostavne tirgonometrijske relacije :

$$\varphi = \tan^{-1}\left(\frac{Dm}{d}\right) \; [rad]$$

Ako u relaciju uvrstimo poznate podatke mjerene za odabranu kameru, možemo izvesti izraz za izračunavanje kuta lopte u odnosu na os robota iz mjerene veličine Xm.

$$\varphi = \frac{180}{\pi}\tan^{-1}\left(\frac{Kc}{320}(Xm - 320)\right) \quad [^\circ]$$

Izvedeni izraz korišten je za računanje kuta objekta u odnosu na os robota prilikom mjerenja oscilacija za određivanje optimalne konstante **Kp**.



Slika 15. Geometrija vidnog polja kamere

4.6. Generiranje udarca

Kada se robot približi lopti na udaljenost manju od 10cm započinje faza udaranja lopte. Udarac se generira tako da se robotu pošalje impuls iznosa 50% maksimalne vrijednosti translacijske brzine u trajanju od 0.2 sekunde. Nakon toga zaustavljaju se sva gibanja robota na 0.5 sekundi kako bismo bili sigurni da se lopta udaljila od robota više od 10cm. Po završetku udarca nastavlja se odvijanje programske petlje kao što je već opisano sve dok se lopta ponovno ne nađe na udaljenosti manjoj od 10cm kada se udarac ponavlja. Kao bi se izbjeglo udaranje drugih prepreka koje se mogu naći na poligonu, za pokretanje udarca provjerava se i vrijednost signala pogreške Xe koji mora biti manji od 10 piksela.

5. IMPLEMENTACIJA PROGRAMA NA RAČUNALO RASPBERRY PI

5.1. Računalo Raspberry Pi

Slika 16. Računalo Raspberry Pi

Raspberry Pi je računalo veličine kreditne kartice, razvijeno od 2006 do 2012 godine u Velikoj Britaniji. Računalo je razvila zaklada Raspberry Pi kao maleno i jeftino računalo namijenjeno učenju programiranja i računalne znanosti. Osnovna komponenta računala je mikrokontroler iz porodice ARM ARM1176JZF-S koji radi na 700MHz. Radna memorija za sada iznosi 512MB. U osnovnoj konfiguraciji na računalo se spaja memorijska kartica sa inačicom operativnog susrava Lynux prilagođenom za ARM arhitekturu Raspbian Wheezy. Inačica je nastala od popularne Lynux distribucije Debian za koju postoji relativno velika baza drivera i ostale podrške za upravljanje periferijama. Za razliku od uobičajenih PC-a i Netbooka, Raspberry Pi uz standardne USB, LAN, HDMI i 3.5mm AUDIO ulaze/izlaze, posjeduje i GPIO (General Purpose Input/Output) modul kojemu je moguće direktno pristupiti. GPIO omogućuje jedostavno spajanje na upravljačke jedinice koračnih motora, servo motora te ostalih aktuatora i senzora. Preko GPIO modula također je moguće

uspostaviti I2C RS232 1-Wire i SPI komunikacije. Pločica se napaja naponom 5V, a kod rada bez puno periferija potrebna je struja od 700mA. Sve navedeno čini Raspberry PI vrlo pogodnim za ugradnju na edukacijske robote kao što je eMIR. [Slika 16] prikazuje pločicu RaspberryPI model B na kojoj se može vidjeti razmještaj konektora.

5.2. Raspberry Pi na robotu eMIR

Raspberry Pi smješten je na pokrovnu ploču robota eMIR. S obzirom da za sada računalo nema ugrađenu kameru za potrebe rješavanja problema loptanja, dodana je vanjska Web kamera. Isto tako na pločici se nalaze samo dva USB porta zbog čega je trebalo dodati uređaj za proširivanje portova, tzv. USB-hub. Napajanje je izvedeno pomoću akumulatora ugrađenog na robotu, no za dulji rad trebalo bi dodati zasebni akumulator kapaciteta oko 2Ah. Komunikacija između upravljačke jedinice i Raspberry Pi računala, uspostavljenja je putem

Slika 17. Raspberry pi na robotu eMIR

Bluetooth RS232 linka, kako bi se provjerilo je li moguće koristiti taj oblik komunikacije. [Slika 17] prikazuje opisanu konfiguraciju računala i kamere na robotu eMIR.

5.3. Modifikacije programa za Raspberry PI

Da bi se opisani program izvršavao na RaspberryPi računalu potrebno je napraviti nekoliko manjih izmjena na programskom kodu koje se odnose na korištenje kamere i uspostavi serijske veze. Upravljanje kamerom odvija se direktno putem paketa Pygame za razliku od windows okruženja gdje se koristi paket VideoCapture. RS 232 veza uspostavlja se slično kao na Netbooku, samo je potrebno nazive uskladiti s praksom na operativnom sustavu Lynux. Kako bismo mogli robotu pristupiti putem Bluetooth-a, potrebno je instalirati paket drivera „Bluez“. Zbog manje radne memorije u eksperimentima je korišten manji zaslon kamere dimenzija 320x240 piksela. S obzirom na nižu rezoluciju zaslona kamere, potrebno je promijeniti regulacijsku konstantu Kp. Eksperimentalno je utvrđeno da sustav najbolje radi s konstantom Kp=0.1.

6. ZAKLJUČAK

U ovom radu opisano je rješavanje problema loptanja robota eMIR. U prvom dijelu rada opisana je strategija gibanja kod loptanja te značajke i način komunikacije sa robotom eMIR. Komunikacija s robotom uspostavljena je serijskom vezom RS-232 putem Bluetooth modula. Ovaj način komunikacije pogodan je za upravljanje pomoću računala jer se informacije razmjenjuju bežično. Program za vođenje robota do lopte izvršavao se na uobičajenom prijenosnom računalu (Netbook). Iako nije za industrijsku primjenu, netbook je korišten kako bi se pokazalo kolike su mogućnosti elektroničkih uređaja koje danas gotovo svi posjedujemo. Čitav program napisan je u programskom jeziku Python, a razvijen u razvojnom okruženju Pyscripter. Python je odabran zbog svojeg Open-Source karaktera, velikog broja vanjskih modula i paketa za upravljanje periferijama i obradu slike te jednostavne implementacije na sve popularnije operativne sustave Lynux. Nedostatak primjene Pythona je otežana izrada .exe datoteka, no u ovom slučaju izrada .exe datoteka nije bila potrebna. Kao glavni senzor za određivanje položaja lopte u odnosu na robota korištena je uobičajena web kamera ugrađena na zaslon prijenosnog računala (Netbooka). Kvaliteta slike i optika kamere pokazali su se dovoljno dobre za izvršavanje zadanog zadatka. Glavni nedostatak web kamere je premala brzina snimanja (frame-rate) zbog čega robot ponekad ne uspije detektirati loptu ako se ona giba prevelikom brzinom. Također web kamera je prilično osjetljiva na intenzitet okolišnog osvjetljenja što utječe na uspješnost detekcije zadane boje. Za regulaciju kutne brzine korišten je regulator P tipa, a regulacijska konstanta određena je eksperimentalno. Za određivanje konstante regulatora tijekom gibanja bilježeno je ponašanje kuta lopte u odnosu na os robota. Mjerenja su ponavljana za različite konstante regulatora te su eksperimenti pokazali da sustav najbolje prati loptu s konstantom regulatora Kp = 0.05. Udaranje lopte ostvareno je naglim povećanjem translacijske brzine robota kada se nađe u neposrednoj brzini lopte. Ovakav način udaranja lopte je odabran jer ne zahtjeva nikakve dodatne mehanizme i aktuatore na robotu. Nakon udarca lopte zaustavljaju se sva gibanja u intervalu od 0.5s kako bi se lopta udaljila od robota. U drugom dijelu rada razvijeni program za loptanje robota pokrenut je na računalu Raspberry Pi koje je ugrađeno na robota eMIR. Komunikacija između robota i RaspberryPi računala također je uspostavljena putem RS232 bluetooth modula. Za izvršavanje na Raspbery Pi računalu rezolucija zaslona kamere smanjena je sa uobičajenih

640x480 piksela na 320x240 nedostatka radne memorije u odnosu na prijenosno računalo Netbook. Isto tako eksperimentalno je utvrđeno da sustav najbolje radi s konstantom regulatora 0.1. U daljnjem radu razvoj bi trebalo temeljiti na računalu Raspberry Pi zbog njegovih velikih mogućnosti i brzog razvoja. Komunikaciju između računala i robota trebalo bi uspostaviti direktno putem GPIO terminala na računalu Raspberry i terminala na upravljačkoj jedinici robota. Za tu svrhu potrebno je modificirati mehaničku konstrukciju robota kako bi računalo sa svojom periferijom bilo zaštićeno od vanjskih utjecaja te ugraditi dodatni akumulator kapaciteta najmanje 2Ah za napajanje računala i periferije. Programski kodovi za izvršavanje na PC računalu s operativnim sustavom Windows te kod za Raspeberry Pi sa operativnim sustavom RaspbianWheezy nalaze se u prilogu.

7. LITERATURA

[1] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. In In International Conference on Robotics and Automation, San Francisco, April 2000.

[2] Freda L., Oriolo, G. Vision based interception of moving target with nonholomonic mobije robot, Robotics and Autonomus systems 55 (2007)

[3] M. Crneković, D. Zorc, Z. Kunica. Research of mobile robot behavior with eMIR, International Conference on Innovative Technologies, IN-TECH 2012

[4] Dušek et. al., Mathematical model of differentially steered mobile robot, International Conference on Process Conrol

[5] Chaumette. A first step toward visual servoing using image moments. In Pro- ceedings of the IEEE Conference on Intelligent Robots and Systems, pages 378–383, October 2002.

[6] Matt Richardson, Shawn Wallace, Getting Started with Raspberry Pi, Publisher: O'Reilly Media / Make, 2012.

Prilog 1. programski kod za izvođenje na PC Windows okruženju :

```
#Učitavanje modula i paketa
from VideoCapture import Device
from PIL import ImageEnhance
import pygame
import pygame.camera
import pygame.draw
from pygame.locals import *
from PIL import Image
import sys
import time
from pylab import*
from math import*
#otvaranje serijske veze :
import serial
RS232 = serial.Serial(9)
RS232.baudrate = 57600
RS232.timeout = 1
res = (640,480)
pygame.init()
camera = Device(0)
camera.setResolution(res[0],res[1])
screen = pygame.display.set_mode((640,480))
pygame.display.set_caption('Webcam')
pygame.font.init()
font = pygame.font.SysFont("Courier",11)
Vt=1
sredina = 20
#otvara log_file
logfile = open('log.txt','w')
#ukljucuje slanje podataka sa senzora
RS232.write('#100100EF/')
time.sleep(0.02)
time.sleep(0.02)
#deklariranje pomoćnih varijabli
x=[]
x2=[]
y=[]
y2=[]
```

```
y3=[]
y4=[]
Xpoz1=[]
Ypoz1=[]
fi1=[]
br=0
sredina=100
#Petlja za izvršavanje zadatka
for i in range(160):
    color=(20,126,20) #novo 20 126 20    staro 69 150 156
    threshold=(30, 60, 30)  #novo 20 50 20 staro 15 20 20
    slika = camera.getImage()
    camshot = pygame.image.frombuffer(slika.tostring(), res, "RGB")
    maska = pygame.mask.from_threshold (camshot,color,threshold)
    koord = maska.centroid()
    # kamera kao senzor
    S=koord[0]
    Km=1
    Xm=S*Km
    # određivanje pozicije
    if (sredina<79) and (sredina>10):
     pom1=((71.0/120.0))/320.0
     pom2=pom1*(sredina+22.0)
     delta=pom2*(Xm-320)
     fi = atan(delta/(sredina+22.0))*(180/pi)
     fi1.append(fi)
     x2.append(br)
    # regulacija
    Kp = 0.05
    Xv = 320
    Xe = Xv-Xm
    Xeplot=Xe/100
    if (Xe<220)and(Xe>-220):
        Vt = '10'
    else:
        Vt='00'
    if sredina<10:
        RS232.write('#013200CD/')
        time.sleep(0.1)
    w = int(-1*round(Kp*Xe))
    w3= int(-1*round(Kp*Xe))
```

```
   #za ispis
   Wisp = w
   if w <0:
      w=256+w
   #pretvara u HEX
   b = hex(w)
   c= b.split('x')
   if w<16 :
    W='0'+c[1]
   else :
    W=c[1]
   # racuna checksum
   Vtrans = int(Vt,16)
   checksum= 256-1-Vtrans-w
   if checksum<0:
      checksum=256+checksum
#checksum to hex
   check = hex(checksum)
   i= check.split('x')
   if checksum<16 :
    Cs='0'+i[1]
   else :
    Cs = i[1]
# salje informaije na robota
   paket = '#01'+Vt+W+Cs+'/'
   paket = paket.swapcase()
   RS232.write(paket)
# cita senzore robota
   Senzori = RS232.readline(size=None,eol='/')
   if len(Senzori)==24:
#Prednji senzor udaljenosti
    sredina = Senzori[1:3]
    sredina = int(sredina,16)
#Omega i V
    brzina = Senzori[15:17]
    try:
     brzina = int(brzina,16)
     if brzina>127:
        brzina =brzina-256
     omega = Senzori[17:19]
     omega = int(omega,16)
```

```
      if omega>127:
         omega=omega-256
      y.append(omega)
      x.append(br)
      y2.append(w3)
      y3.append(sredina)
      y4.append(Xeplot)
#pise u logfile
      Omega=str(omega)
      Brzina=str(brzina)
      er=str(Xe)
      Wref=str(Wisp)
      Sredina=str(sredina)
      logfile.write(er+','+Wref+'  '+paket+'  '+Senzori+'  '+Brzina+','+Omega+','+Sredina+'\n')
     except ValueError :
        brzina = 0
        omega=0
#oznacava_objekt
    circ = pygame.draw.circle(screen,(200,10,20),koord,10,0)
#crta ciljnik
    line1=pygame.draw.line(screen,(200,10,20),(320,0),(320,480),2)
    pygame.display.flip()
    for event in pygame.event.get():
       if event.type == pygame.QUIT: sys.exit()
    screen.blit(camshot, (0,0))
    pygame.display.flip()
    RS232.flushOutput()
    RS232.flushInput()
    time.sleep(0.1)
    br=br+1
#zaustavlja robota
RS232.write('#010000FF/')
# zatvaranje komunikacijskih kanala
RS232.close()
logfile.close()
#crta dijagrame servisnih veličina
plot(x2,fi1,'bo-')
plot(x,y2,'bo-')
plot(x,y3,'go-')
plot(x,y4,'yo-')
show()
```